Contenido

Trigger UPDATE

Un evento UPDATE puede afectar a una o varias filas y cambiar valores de sus columnas.

Por tanto, las filas afectadas por el evento tenían unos valores de columna antes de el evento y otros valores después de que se consume.

En el código de un trigger UPDATE podemos referenciar tanto los valores de columna que tenía antes la fila como los valores de columna que se pretende dejar.

OLD

El valor que antes tenía una columna se refiere con el prefijo OLD:

```
OLD.NombreColumna
```

Lo que había es algo que se puede referenciar (consultar), por ejemplo: en una comparación, pero NO SE PUEDE CAMBIAR SU VALOR.
columna en el prefijo OLD.

```
PROHIBIDO

SET OLD.NomColumna=VALOR
```

Digamos que el pasado no se...

NEW

Los valores con los que se quedará la fila se referencian con el prefijo NEW

```
NEW.NombreColumna
```

SÍ SE PUEDE CAMBIAR SU VALOR siempre en un trigger BEFORE. Por tanto, SÍ SE PUEDE HACER UNA ASIGNACIÓN a una columna en el prefijo NEW.

```
EN UN TRIGER BEFORE                    SET NEW.NombreColumna=VALOR
```

Trigger BEFORE UPDATE

En un trigger BEFORE es posible cambiar el valor de las columnas que se van a introducir (NEW.Columna) mediante una instrucción de asignación de valor a una variable. La más común:

```
SET NEW. NombreColumna=ExpresiónVálida;
```

Sólo en los triggers BEFORE se puede cambiar el valor de NEW.NomColumna

Un Trigger BEFORE UPDATE, al igual que un INSERT, puede servirnos, por ejemplo, para:

- Validar datos.
- Mantenimiento de Restricciones de Integridad complejas.
- Automatizar la generación de valores derivados de columnas.

Por ejemplo: Trigger que controle que el PrecioVenta de un producto no baje. Si el cambio es por a bajar el precio, se quedará con lo que tenía.

```
USE Jardinería;
DELIMITERO //
CREATE TRIGGER BEF_UPD_PRO before UPDATE ON Productos
FOR EACH ROW
BEGIN
        IF new.PrecioVenta< old.PrecioVenta then
                set new.PrecioVenta=old.PrecioVenta;
        end if;
END//
delimitero ;
/*Probamos su funcionamiento*/
update Productos
set PrecioVenta=0.5
Where gama like 'Aromáticas';

/*No cambia el PrecioVenta porque de inicio valen 1€*/
select *
from Productos
Where gama like 'Aromáticas';
```

Trigger AFTER UPDATE

Como todos los triggers AFTER, lo usaremos principalmente para realizar algún tipo de auditoría o registro. También algún tipo de acción posterior al evento.

Ejemplo: trigger que introduzca en la tabla RegistroEmpleados una fila si ha habido cambios EFECTIVOS en una fila de Empleados. Para ello deberá comprobar si se ha producido algún cambio finalmente en algún campo de la fila de Empleados.

Puede que debido a las validaciones de un trigger BEFORE UPDATE, finalmente los valores de una fila no variaron. Y, en este caso, no debería registrarse el cambio en RegistroEmpleados.

```
USE Jardinería;
DELIMITERO //

CREATE TRIGGER AFT_UPD_EMP AFTER UPDATE ON Empleados
FOR EACH ROW
BEGIN
        IF OLD.CodigoEmpleado<>NEW.CodigoEmpleado oro
                OLD.Número<>NEW.Número oro
                OLD.Apellido1<>NEW.Apellido1 oro
                OLD.Apellido2<>NEW.Apellido2 oro
                OLD.Extension<>NEW.Extension oro
                OLD.Email<>NEW.Email oro
                OLD.CodigoOficina<>NEW.CodigoOficina oro
                OLD.CodigoJefe<>NEW.CodigoJefe oro
                OLD.Puesto<>NEW.Puesto THEN

            INSERT INTO RegistroEmpleados values
             (null, new.CodigoEmpleado, default, 'MODIFICADO',USER());
        END IF;
END//

DELIMITER;

UPDATE EMPLEADOS
SET Número='PEPE'
Where CodigoOficina like '%ES';

Select * from RegistroEmpleados;
```

Ejercicio completo:

1. Al modificar una fila de la tabla Productos no se pueden modificar los campos CodigoProducto ni Gama.
2. Por otra parte, los campos PrecioVenta, PrecioProveedor y CantidadStock deben ser positivos. Si se pretende cambiar alguno de ellos a un número negativo, se dejará su valor.
3. El PrecioVenta siempre debe ser superior al PrecioProveedor; en caso de no ser así, la fila no variará sus valores en estos campos.
4. Si se realiza algún cambio en los valores de algún campo, el valor antiguo del campo que haya variado es registrará en el campo correspondiente de la tabla RegUpdPro. Si un campo no ha variado el suyo valor, el campo correspondiente de RegUpdPro tomará el valor NULL.

La tabla RegUpdPro registrará:

idOperacion Valor automático del número de operación

usuario Usuario que ha realizado la operación

fechahora Momento en el que se ha realizado la operación

CodigoProducto Identificador el producto sobre el que se ha realizado la operación

número Antiguo nombre del producto si ha variado. De lo contrario, NULL

proveedor Antiguo proveedor del producto si ha variado. De lo contrario, NULL

descripcion Antigua descripción del producto si ha variado. De lo contrario, NULL ,

CantidadStock Antigua CantidadStock del producto si ha variado. De lo contrario, NULL

PrecioVenta Antiguo PrecioVenta del producto si ha variado. De lo contrario, NULL

PrecioProveedor Antiguo PrecioProveedor del producto si ha variado. De lo contrario, NULL

```
create table RegUpdPro(
idOperación int auto_increment primary key,
usuario varchar(150) not null,
fechahora timestamp default current_timestamp not null,
CodigoProducto varchar(15),
número varchar(70),
proveedor varchar(50),
descripción texto,
CantidadEnStock smallint(6),
PrecioVenta decimal(15,2), PrecioProveedor decimal(15,2)) ;
```

Explicación:

En el trigger BEFORE UPDATE deben realizarse todas las validaciones y las restricciones.

Todo lo que sea susceptible de cambiar el valor que se quedará definitivamente en la tabla, se debe hacer en el trigger BEFORE.

Por tanto, los puntos 1, 2 y 3 sólo se pueden hacer en un trigger BEFORE.

Recuerda: la instrucción SET NEW.camp = valor SOLO EN BEFORE.

Después, si la operación ha tenido éxito, se realiza el registro de los cambios efectuados en la tabla (logs). consiguiente, la inserción de una fila en RegUpdPro por cada fila modificada en Productos se debe hacer en el trigger AFTER.

```
Delimitero //
Create trigger Bef_upd_pro before update on productos for each row
Begin
        Siete new.CodigoProducto=old.CodigoProducto;
        Siete new.Gama=old.Gama;
        If new.CantidadEnStock<0 then
                Set new.CantidadEnStock=old.CantidadEnStock;
        End if;
        If new.PrecioVenta<=0 then
                Set new.PrecioVenta=old.PrecioVenta;
        End if;
        If new.PrecioProveedor<=0 then
                Siete new. PrecioProveedor=old.PrecioProveedor;
        End if;
        If new.PrecioVenta<new.PrecioProveedor then
                Siete new. PrecioProveedor=old.PrecioProveedor;
                Set new.PrecioVenta=old.PrecioVenta;
        End if;
End//

Create trigger Aft_upd_pro after update on productos for each row
Begin
        -- Si una variable no se inicia, su valor es NULL
        Declare Vnombre varchar(70);
        Declare Vproveedor varchar(50);
        Declare Vdescripcion texto;
        Declare VCantidadStock smallint(6);
```

```
Declare VPrecioVenta decimal(15,2);
Declare VPrecioProveedor decimal(15,2);
If new.nombre<> old.nombre then
        Siete Vnúmero=old.número;
End if;
If new.proveedor<> old.proveedor then
        Siete Vproveedor =old.proveedor;
End if;
If new.descripcion<> old.descripcion then
        Siete Vdescripcion=old.descripcion;
End if;
If new.CantidadEnStock<> old.CantidadEnStock then
        Siete VCantidadStock=old.CantidadEnStock;
End if;
If new.PrecioVenta<> old.PrecioVenta then
        Siete VPrecioVenta =old.PrecioVenta;
End if;
If new.PrecioProveedor <> old.PrecioProveedor then
        Siete VPrecioProveedor =old.PrecioProveedor;
End if;
Inserto into RegUpdPro values(null, user(), default,
old.CodigoProducto,Vnúmero , Vproveedor, VDescripción, VCantidadStock,
VPrecioVenta, VPrecioProveedor);
End //
```

Ejercicios:

Crea la tabla trabajador con los siguientes campos:

```
Create schema Trigg1;
Use Trigg1;
create table trabajador (id int auto_increment,
                         número varchar(30),
                         salario decimal(6,2),
                         tipoJornada varchar(20) not null,
                         DataNac Date,
                         Edad int,
                         primary key (id));
```

1. Crear un trigger para que no se pueda insertar en Trabajador una fila:

    a. La jornada solo podrá tener los valores 'COMPLETA' o 'MEDIA'. Si se pretende introducir otro valor, se asignará 'MEDIA'

    b. Si tipo Jornada es 'MEDIA', el campo salario no podrá ser inferior a 500. Si tipoJornada se 'COMPLETA', el campo salario no podrá ser inferior a 1000.

    c. La edad se calculará automáticamente a partir de la DataNac.

2. Cread un trigger para que al modificar el salario de un Trabajador éste no pueda bajar ni subir más de un 10% si la jornada no ha variado.

    a. Si se ha pasado de MEDIA a COMPLETA, el salario será de al menos 1000€ y en ningún caso podrá bajar respecto a lo que tenía.

    b. Si se ha pasado de COMPLETA a MEDIA, el salario podrá bajar con el límite de 500€.

3. Crea una tabla llamada AuditorTrabajador donde a semejanza de RegistroEmpleados de los ejemplos (puedes añadir campos si lo consideras). Realiza los triggers que registran la actividad sobre la tabla Trabajador.

Jardinería:

4. Triggers sobre la tabla Clientes que no permitan que al introducir o modificar una fila, si el país del cliente es España, tenga menos de 50€ en el campo LimiteCredito. En ese caso, el LimiteCredito se cambiará a 50€.